Actividad 04

Alumno: José Osvaldo Farfán de León

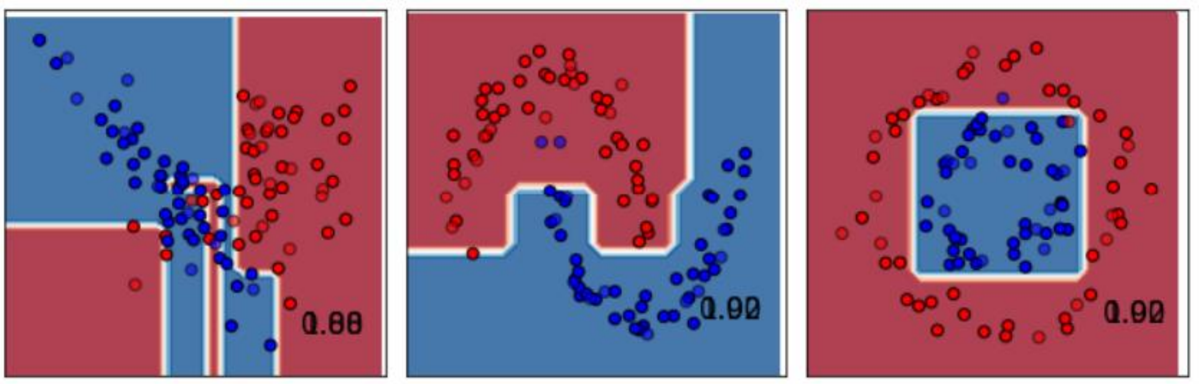
Materia: Seminario de IA ll

Profesor: Carlos Alberto Villaseñor Padilla

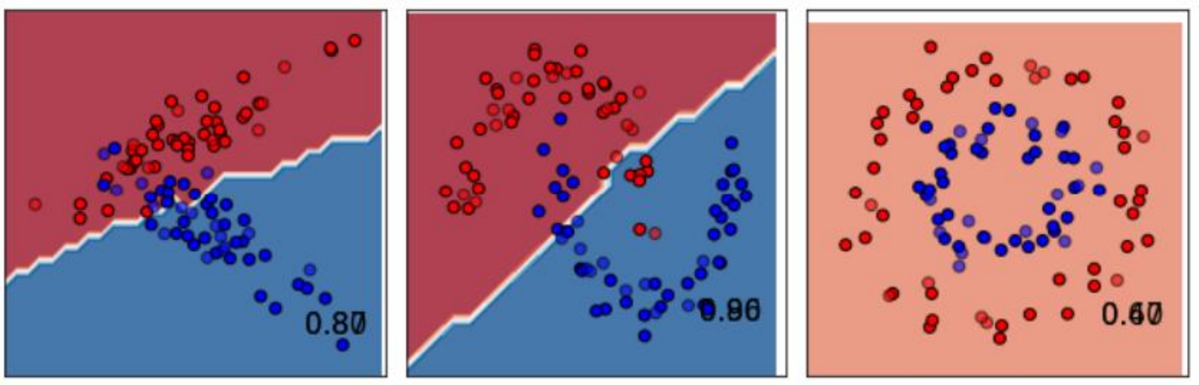
Sección: “D04”

Comparación de clasificadores no lineales

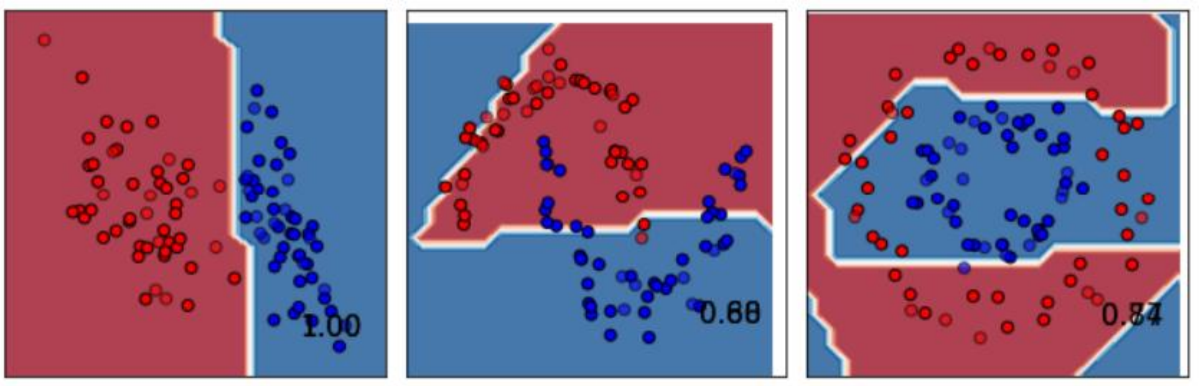
Decision Tree



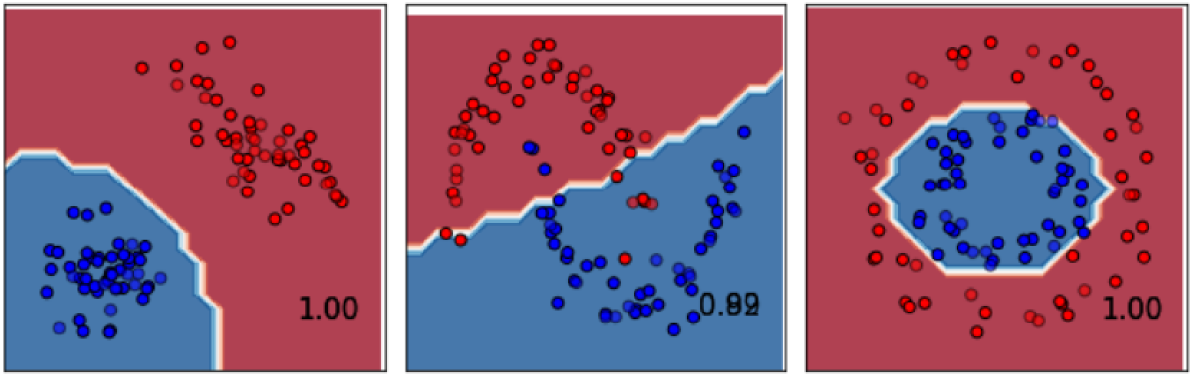
KNN

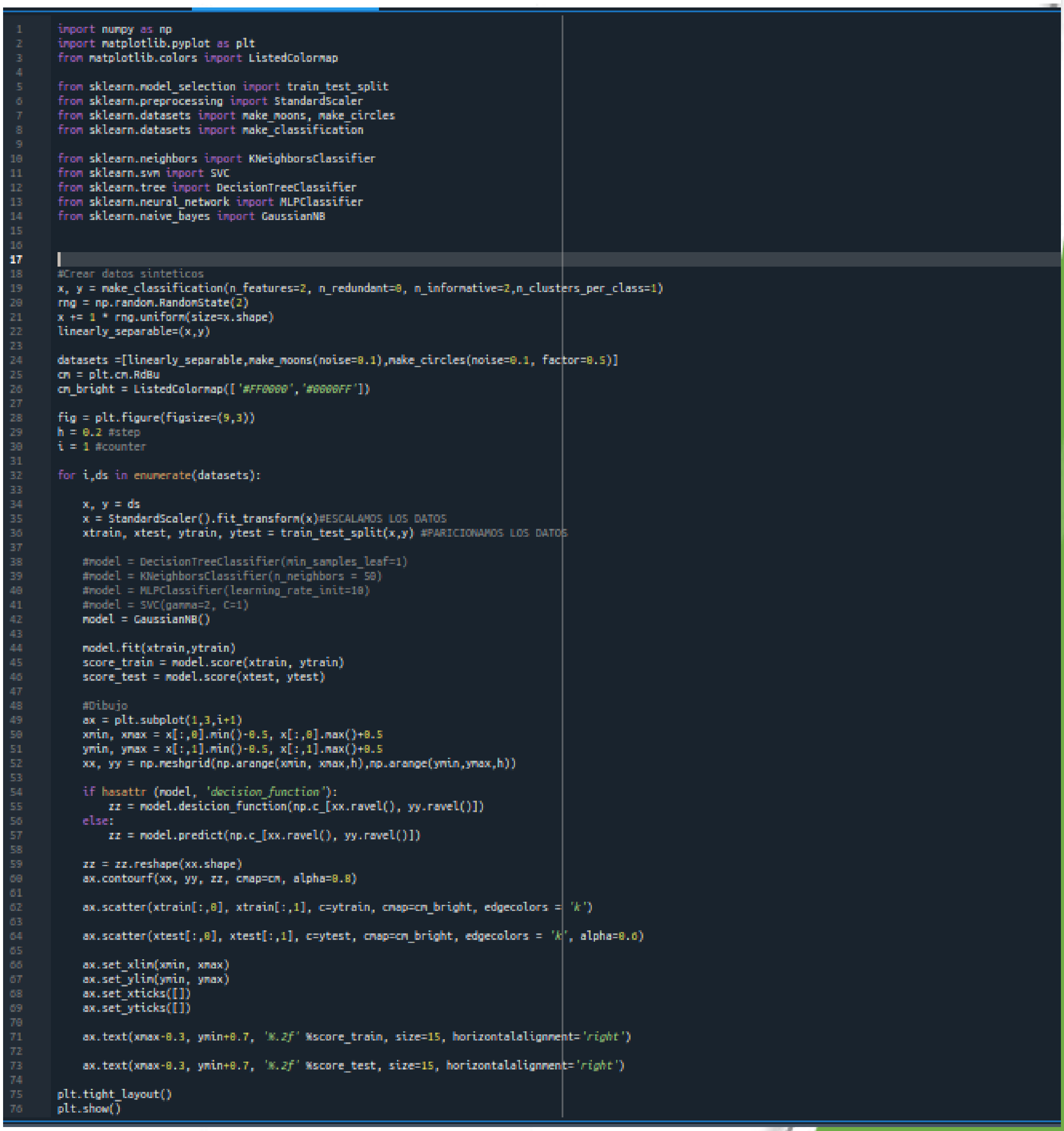


MLP



Gaussian





```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles
from sklearn.datasets import make_classification

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB



#Crear datos sinteticos
x, y = make_classification(n_features=2, n_redundant=0, n_informative=2,n_clusters_per_class=1)
rng = np.random.RandomState(2)
x += 1 * rng.uniform(size=x.shape)
linearly_separable=(x,y)

datasets =[linearly_separable,make_moons(noise=0.1),make_circles(noise=0.1, factor=0.5)]
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000','#0000FF'])

fig = plt.figure(figsize=(9,3))
h = 0.2 #step
i = 1 #counter

for i,ds in enumerate(datasets):

    x, y = ds
    x = StandardScaler().fit_transform(x)#ESCALAMOS LOS DATOS
    xtrain, xtest, ytrain, ytest = train_test_split(x,y) #PARICIONAMOS LOS DATOS

    #model = DecisionTreeClassifier(min_samples_leaf=1)
    #model = KNeighborsClassifier(n_neighbors = 50)
    #model = MLPClassifier(learning_rate_init=10)
    #model = SVC(gamma=2, C=1)
    model = GaussianNB()

    model.fit(xtrain,ytrain)
    score_train = model.score(xtrain, ytrain)
    score_test = model.score(xtest, ytest)

    #Dibujo
    ax = plt.subplot(1,3,i+1)
    xmin, xmax = x[:,0].min()-0.5, x[:,0].max()+0.5
    ymin, ymax = x[:,1].min()-0.5, x[:,1].max()+0.5
    xx, yy = np.meshgrid(np.arange(xmin, xmax,h),np.arange(ymin,ymax,h))

    if hasattr (model, 'decision_function'):
        zz = model.desicion_function(np.c_[xx.ravel(), yy.ravel()])
    else:
        zz = model.predict(np.c_[xx.ravel(), yy.ravel()])

    zz = zz.reshape(xx.shape)
    ax.contourf(xx, yy, zz, cmap=cm, alpha=0.8)

    ax.scatter(xtrain[:,0], xtrain[:,1], c=ytrain, cmap=cm_bright, edgecolors = 'k')

    ax.scatter(xtest[:,0], xtest[:,1], c=ytest, cmap=cm_bright, edgecolors = 'k', alpha=0.6)

    ax.set_xlim(xmin, xmax)
    ax.set_ylim(ymin, ymax)
    ax.set_xticks([])
    ax.set_yticks([])

    ax.text(xmax-0.3, ymin+0.7, '%.2f' %score_train, size=15, horizontalalignment='right')

    ax.text(xmax-0.3, ymin+0.7, '%.2f' %score_test, size=15, horizontalalignment='right')

plt.tight_layout()
plt.show()
```